Integer programming for the generalized high school timetabling problem

Kristiansen, Simon; Sørensen, Matias; Stidsen, Thomas Riis

Published in:
Journal of Scheduling

Link to article, DOI:
10.1007/s10951-014-0405-x

Publication date:
2015

Document Version
Peer reviewed version

Noname manuscript No.
(will be inserted by the editor)

Integer Programming for the Generalized (High) School Timetabling Problem

Simon Kristiansen · Matias Sørensen · Thomas R. Stidsen

Abstract Recently the XHSTT format for (High) School Timetabling was introduced, which provides a uniform way of modeling problem instances and corresponding solutions. The format supports a big variety of constraints, and currently 38 real-life instances from 11 different countries are available. Thereby the XHSTT format serves as a common ground for researchers within this area. This paper describes the first exact method capable of handling an arbitrary instance of the XHSTT format. The method is based on a Mixed-Integer linear Programming (MIP) model, which is solved in two steps with a commercial general-purpose MIP solver. Computational results show that our approach is able to find previously unknown optimal solutions for 2 instances of XHSTT, and proves optimality of 4 known solutions. For the instances not solved to optimality, new non-trivial lower bounds were found in 11 cases, and new best-known solutions were found in 9 cases. Furthermore the approach is shown to be competitive with the finalist of Round 2 of the International Timetabling Competition 2011.

Keywords High School Timetabling · XHSTT · Integer Programming

S. Kristiansen (E-mail: sikr@dtu.dk) · M. Sørensen (E-mail: msso@dtu.dk) · T.R. Stidsen
Section of Operations Research, Department of Management Engineering, Technical University of Denmark
Building 426, Produktionstorvet, DK-2800 Kgs. Lyngby, Denmark

S. Kristiansen · M. Sørensen
MaCom A/S, Vesterbrogade 48, 1., DK-1620 Copenhagen V, Denmark

1 Introduction

The problem of scheduling lectures to time slots and/or resources at high schools is known as the *High School Timetabling* (HST) problem. This is an important problem for high schools in many countries, and a large amount of different solution approaches have been proposed, see the survey Schaerf (1999).

It is well recognized that the specifications of the HST problem varies significantly depending on the country of which the problem originates, and that the problem in general is hard to solve. With the introduction of the XHSTT format (Post et al, 2012a), a large number of instances from various origins became publicly available in standardized form. The format is based on the *Extensible Markup Language* (XML) standard, and all instances are available online (Post, 2013a). One purpose of the format is to serve as a common testbed for *school timetabling*, in an attempt to promote research within this area. In this context, "school timetabling" denotes the area covering high school timetabling and *university course timetabling*, as the format has also been shown capable of modeling some instances of the latter problem (see Kingston (2013a) for an overview of educational timetabling problems).

This paper describes the first exact method capable of handling an arbitrary instance of the XHSTT format. The method is based on a *Mixed-Integer linear Programming* (MIP) model, which is solved in two steps with a commercial general-purpose MIP solver. Computational results are performed for all the real-life instances currently available. Thereby we are able to find previously unknown optimal solutions, and prove optimality of already known solutions.

To the best of our knowledge, all previous solution methods for the XHSTT format have been heuristic in nature. Therefore no proof of optimality has been made

for any instance, except for those instances where a solution with objective value 0 is known, since 0 is a trivial lower bound for any XHSTT instance. The obvious advantage of *Integer Programming* (IP) over heuristic methods is the capability to issue certificates of optimality. Therefore it is remarked that a big advance within general-purpose MIP solvers has happened in recent years, see e.g. Bixby (2012). Even though the MIP we will present is inevitable complex in nature, it will be shown that it can be used to find optimal solutions for several instances of the XHSTT archive `ALL_INSTANCES`. For those instances where an optimal solution cannot be found, we are able to show a non-trivial lower bound on optimum in the majority of cases. These are significant results for high school timetabling in general.

The outline of this paper is as follows. Section 2 presents related literature. Section 3 presents the MIP model of XHSTT. Section 4 describes computational results. Finally, Section 5 concludes and describes future research possibilities.

2 Related Literature

The Third International Timetabling Competition (ITC2011) considered the HST Problem, based on instances of the XHSTT format (Post et al, 2012b). Four teams were part of the final round: The overall winner (Team Goal) used Simulated Annealing and Iterated Local Search to perform local search around a generated initial solution (Fonseca et al, 2012). Participant from the University of Nottingham (HySTT) used a method based on Hyper-heuristics (Kheiri et al, 2012). Team Lectio used Adaptive Large Neighborhood Search (ALNS) (Sørensen et al, 2012). Romrös and Homberger (2012) (Team HFT) used an Evolutionary Algorithm. The results of the competition can be found at the official homepage of ITC2011 (Post (2013b)).

Pimmer and Raidl (2013) describe a 'timeslot-filling' heuristic for XHSTT, which iteratively fills selected timeslots with sets of events. Two state-of-the-art solutions were found for instances of the archive `XHSTT-2012`. Ter Braak (2012) presents a Hyper-heuristic and several other heuristics for the XHSTT.

Valouxis et al (2012) describe a two-phase approach based on MIP used to solve the Greek case of the HST problem. This includes two instances which are part of the XHSTT project, and which were both solved to optimality (solutions were found with an objective value of 0).

In terms of Integer Programming and HST problems not based on XHSTT, the following contribution are mentioned: Santos et al (2012) present a *Column Generation* approach for establishing bounds for a set of datasets originating from Brazil. Birbas et al (2009) present an approach for Greek datasets where the *Shift Assignment Problem* is solved first, and the timetable is constructed on the basis on these work-shifts for teachers. The paper of Sørensen and Stidsen (2013) describes a complex MIP of the Danish case of high school timetabling, and establishes computational results for 100 real-life instances. Avella et al (2007) present an algorithm based on Very Large-Scale Neighborhood search where the neighborhood is explored by a MIP, for Italian cases of high school timetabling.

3 Problem Description and a Mixed Integer Programming Formulation

In this section a brief description of the specifications of the XHSTT format is given, and a MIP model is formulated. The entire documentation of XHSTT is available at Post (2013a). We do not intend to describe all properties of the format, but only those necessary to formulate the MIP.

An instance of XHSTT consists of *times* (denoted $\mathcal{T}$ in the following), *time groups* (denoted $\mathcal{TG}$), *resources* (denoted $\mathcal{R}$), *events* (denoted $\mathcal{E}$), *event groups* (denoted $\mathcal{EG}$) and *constraints* (denoted $\mathcal{C}$). An event $e \in \mathcal{E}$ has a duration $D_e \in \mathbb{N}$, and a number of *event resources* which we each denote $er \in e$. An event resource defines the requirement of the assignment of a resource to the event, and this resource can be specified to be preassigned. If the resource is not preassigned, a resource of proper *type* must be assigned. Furthermore an event resource er can undertake a specific role_{er}, which is used to link the event resource to certain constraints.

It is the job of any solver for XHSTT to decide how each event should be split into *sub-events*. A sub-event se is defined as a fragment of a specific event $e \in \mathcal{E}$, has a duration $D_{se} \leq D_e$, and inherits the requirement of resources defined by the event, such that each sub-event has the exact same resource requirements as the event. Let $\mathcal{SE}$ denote the entire set of sub-events, and let $se \in e$ specify that sub-event se is part of event e. The total duration of all sub-events for event $e \in \mathcal{E}$ in a solution cannot exceed D_e. In our model formulation we create the 'full set' of sub-events with different lengths, i.e. all possible combinations of sub-events for a given event can be handled. E.g. if an event has duration 4, the set of sub-events for this event has the respective lengths $1, 1, 1, 1, 2, 2, 3$ and 4. As a constraint it is then specified that the summed duration of the *active* sub-events in a solution must equals 4. A sub-event is active

if it is assigned a starting time or a non-preassigned resource. An active sub-event is analogous to the concept of *solution events* defined in the XHSTT documentation.

The times $\mathcal{T}$ are ordered in chronological order, and we let $\rho(t)$ denote the index number of time t in $\mathcal{T}$. A time group $\mathcal{TG}$ defines a set of times, and we let $t \in tg$ denote that time t is part of time group tg.

Each constraint $c \in \mathcal{C}$ is of a specific type, and the set $\mathcal{C}$ can contain several constraints of the same type. Each constraint applies to certain events, event groups or resources, and penalizes certain characteristics of the timetable for these entities.

The following notation shorthand is made: By the notions $e \in c$, $r \in c$, $eg \in c$ we denote that constraint $c \in \mathcal{C}$ applies to event $e \in \mathcal{E}$, resource $r \in \mathcal{R}$, and event group $eg \in \mathcal{EG}$, respectively.

The set of resources and times are both extended with a dummy-index, denoted the *dummy-resource* r_D and the *dummy-time* t_D, respectively. These are necessary to ensure feasibility as we create all combinations of sub-events for each event, and not all of these can be assigned a time or the required resources without the duration of the active sub-events exceeding the duration of the event. Thereby these dummy-elements in fact represent that an event resource is *not* assigned a resource, and that a sub-event is *not* assigned a starting time, respectively.

3.1 Objective Function

Each XHSTT constraint penalizes timetables with certain characteristics, which contributes to the objective function of the MIP. Each constraint $c \in \mathcal{C}$ has a set of *point-of-applications* (indexed by $p \in c$). With each point-of-application is associated a set of *deviations* (indexed by $d \in p$), and each deviation has a non-negative cost associated with it. How this cost is calculated depends on the constraint type. The cost of a point-of-application is found on basis of the cost of the deviations, and is influenced by an indication on the constraint whether the constraint is a *hard* or a *soft* constraints, the *weight* of the constraint ($\omega_c \in \mathbb{N}$) and an indication of which *CostFunction* to use. For each constraint $c \in \mathcal{C}$ we let the variable $s_{c,p,d} \in \mathbb{N}$ be the penalty value of the deviation $d \in p$ of the point-of-application $p \in c$. The set of point-of-applications and deviations should be understood in an abstract context; E.g. depending on the type of the constraint, a point-of-application could be an event, a resource, etc., and likewise for the deviations.

The objective of a solution consists of a value for both the hard constraints (denoted *hard cost*) and a value for the soft constraints (denoted *soft cost*). Usually the objective value of a solution is written as (hard cost, soft cost). The hard cost always takes priority over the soft cost, i.e. solutions are first ranked on their hard cost, and secondly on the soft cost. How this type of objective function is handled in context of a MIP is described in Section 3.4.

The cost of a constraint $c \in \mathcal{C}$ which contains slack variable $s_{c,p,d}$ is denoted $f(s_{c,p,d})$,

$$f(s_{c,p,d}) = \omega_c \cdot \text{CostFunction}(s_{c,p,d}) \tag{1}$$

Five different types of CostFunction are allowed. The most trivial one is `Sum`, which simply sums the penalty value of all deviations for all point-of-applications. In the following each CostFunction is formulated in linear terms. Let the variable $\text{obj}_c \in \mathbb{N}_0$ denote the value of the of the CostFunction of constraint $c \in \mathcal{C}$.

- **Sum:** Sum the deviations.

$$\text{obj}_c = \sum_{p \in c, d \in p} s_{p,d,c} \ \forall c \in \mathcal{C} \tag{2}$$

- **SumSquare:** Sum the squares of the deviations.
  To cope with this non-linear cost function, the variable $s_{c,p,d,i} \in \{0,1\}$ is introduced, which takes value 1 if the deviation $d \in p$ of the point of application $p \in c$ of constraint $c \in \mathcal{C}$ has the penalty $i \in \mathcal{I}$, and 0 otherwise. The objective value is defined as follows:

$$\text{obj}_c = \sum_{p \in c, d \in p, i \in \mathcal{I}} i^2 \cdot s_{c,p,d,i} \ \forall c \in \mathcal{C} \tag{3}$$

  However we also need to make sure that only a single integer value is selected,

$$\sum_{i \in \mathcal{I}} s_{c,p,d,i} = 1 \ \forall c \in \mathcal{C}, p \in c, d \in p \tag{4}$$

  The amount of elements in the set $\mathcal{I}$ determines the maximum possible penalty for a deviation, and thereby influence the maximum possible penalty for a constraint. To maintain optimality of the model, it is therefore important that the size of $\mathcal{I}$ is selected sufficiently large. This is elaborated in Section 4.
- **SquareSum:** Square the sum of deviations.
  The binary slack variable $u^{\text{squaresum}}_{c,p,j} \in \{0,1\}$ is introduced, which takes value 1 if the point of application $p \in c$ of constraint $c \in \mathcal{C}$ has the deviation $j \in \mathcal{J}$, and 0 otherwise.

$$\text{obj}_c = \sum_{p \in c, j \in \mathcal{J}} j^2 \cdot u^{\text{squaresum}}_{c,p,j} \ \forall c \in \mathcal{C} \tag{5}$$

$$\sum_{j \in \mathcal{J}} u^{\text{squaresum}}_{c,p,j} = 1 \ \forall c \in \mathcal{C}, p \in c \tag{6}$$

$$\sum_{d \in p} s_{p,d,c} = \sum_{j \in \mathcal{J}} j \cdot u^{\text{squaresum}}_{c,p,j} \ \forall c \in \mathcal{C}, p \in c \tag{7}$$

Table 1 Different constraint types in the XHSTT format (Post et al, 2012b)

Constraint	Description
Assign Resource	Event resource should be assigned a resource
Assign Time	Event should be assigned a time
Split Events	Event should split into a constrained number of sub-events
Distribute Split Events	Event should split into sub-events of constrained durations
Prefer Resources	Event resource assignment should come from resource group
Prefer Times	Event time assignment should come from time group
Avoid Split Assignments	Set of event resources should be assigned the same resource
Spread Events	Set of events should be spread evenly through the cycle
Link Events	Set of events should be assigned the same time
Order Events	Set of events should be ordered
Avoid Clashes	Resource's timetable should not have clashes
Avoid Unavailable Times	Resource should not be busy at unavailable times
Limit Idle Times	Resource's timetable should not have idle times
Cluster Busy Times	Resource should be busy on a limited number of days
Limit Busy Times	Resource should be busy a limited number of times each day
Limit Workload	Resource's total workload should be limited

Like the set $\mathcal{I}$, the size of the set $\mathcal{J}$ must be selected sufficiently large to maintain optimality, see Section 4.

- **SumStep:** This penalizes by the number of positive deviations, irrespective of their value.
  The binary variable $u^{\text{sumstep}}_{c,p,d} \in \{0,1\}$ is introduced, which takes value 1 iff $s_{c,p,d} > 0$ for constraint $c \in \mathcal{C}$, point-of-application $p \in c$ and deviation $d \in p$, and 0 otherwise.

$$\text{obj}_c = \sum_{p \in c, d \in p} u^{\text{sumstep}}_{c,p,d} \ \forall c \in \mathcal{C} \tag{8}$$

$$M \cdot u^{\text{sumstep}}_{c,p,d} \geq s_{c,p,d} \ \forall c \in \mathcal{C}, p \in c, d \in p \tag{9}$$

  where $M \in \mathbb{N}$ is some sufficiently large number.
- **StepSum:** This CostFunction penalizes by investigating whether the constrain contains at least one positive deviation. If this is not the case, the penalty is 0.
  The binary variable $u^{\text{stepsum}}_{c} \in \{0,1\}$ is introduced, which takes value 1 if there exists at least one positive deviation for the constraint $c \in \mathcal{C}$, and 0 otherwise.

$$\text{obj}_c = u^{\text{stepsum}}_{c} \ \forall c \in \mathcal{C} \tag{10}$$

$$M \cdot u^{\text{stepsum}}_{c} \geq s_{c,p,d} \ \forall c \in \mathcal{C}, p \in c, d \in p \tag{11}$$

  where $M \in \mathbb{N}$ is some sufficiently large number.

3.2 Mixed-Integer Programming Formulation

In this section the variables and the constraints of the MIP are described. As a basis for our approach is the variable $x_{se,t,er,r} \in \{0,1\}$, which takes value 1 if sub-event $se \in \mathcal{SE}$ has been assigned time $t \in \mathcal{T}$ as starting time and resource $r \in er$ is assigned to event resource $er \in se$, and 0 otherwise. To simplify notation, and to reduce the amount of non-zeros in the MIP, three auxiliary variables are introduced which all 'inherits' their values directly from $x_{se,t,er,r}$. Let the binary variable $y_{se,t} \in \{0,1\}$ take value 1 if sub-event $se \in \mathcal{SE}$ has been assigned time $t \in \mathcal{T}$ as starting time, and 0 otherwise. The variable $v_{t,r} \in \mathbb{N}_0$ denotes the number of times resource r is used in time t by any set of sub-events. Let variable $w_{se,er,r} \in \{0,1\}$ take value 1 if sub-event $se \in \mathcal{SE}$ is assigned resource $r \in \mathcal{R}$ for event resource er, and 0 otherwise.

3.2.1 Base Constaints

Besides all the constraints described in the specifications of the XHSTT, some basic constraints are needed to ensure feasibility. First of all we need to make sure that a sub-event is assigned only one starting time and that the number of resource assigned is exactly the same as the number of event resources of the event.

$$\sum_{t \in \mathcal{T}, r \in \mathcal{R}} x_{se,t,er,r} = 1 \ \forall se \in \mathcal{SE}, er \in se \tag{12}$$

The following constrains variable $y_{se,t}$, and together with (12) ensures that a sub-event is not spread across multiple times. We denote by $|er|_{se \in e}$ the amount of event resources for event e of sub-event se.

$$\sum_{er \in se, r \in \mathcal{R}} x_{se,t,er,r} = |er|_{se \in e} \cdot y_{se,t} \ \forall se \in \mathcal{SE}, \ t \in \mathcal{T} \tag{13}$$

The link to variable $v_{t,r}$ is shown in eq. (15). For time $t \in \mathcal{T}$ and $se \in \mathcal{SE}$ is found the set of possible starting-times for se which will cause resource $r \in \mathcal{R}$ to be used in time $t \in \mathcal{T}$. Let the set $T^{\text{start}}_{se,t} \subseteq \mathcal{T}$ be the set of times which sub-event se lies in if it is assigned starting time t, i.e.

$$T^{\text{start}}_{se,t} = \{t' \in \mathcal{T} \setminus t_D \mid \rho(t) - D_{se} + 1 \leq \rho(t') \leq \rho(t)\} \tag{14}$$

$$\sum_{se \in \mathcal{SE}, er \in se, t' \in T^{\text{start}}_{se,t}} x_{se,t',er,r} = v_{t,r} \ \forall t \in \mathcal{T} \setminus t_D, r \in \mathcal{R} \tag{15}$$

The link to variable $w_{se,er,r}$ looks as follows:

$$\sum_{t \in \mathcal{T}} x_{se,t,er,r} = w_{se,er,r} \ \forall se \in \mathcal{SE}, er \in se, r \in er \tag{16}$$

A sub-event cannot be assigned a start time if there is not enough continuous times after the start time to fulfill the duration, ensured by the constraint:

$$y_{se,t} = 0 \ \forall se \in \mathcal{SE}, t \in \mathcal{T} \setminus t_D, \rho(t) + D_{se} - 1 > |\mathcal{T}| \tag{17}$$

Active Sub-events

As we create all possible sub-events for a given event, only a subset of these should be active in the final solution. The binary variable $u_{se} \in \{0,1\}$ takes value 1 if sub-event $se \in \mathcal{SE}$ is active and 0 otherwise. Recall that a sub-event is active if its assigned a starting time, or if is assigned at least one non-preassigned resource. Let the parameter $PA_{er} \in \{0,1\}$ take value if event resource er has a preassigned resource, and 0 otherwise. The following constraints are imposed.

$$\sum_{r \in er \setminus r_D} w_{se,er,r} \leq u_{se} \ \forall se \in \mathcal{SE}, er \in se, PA_{er} = 0 \tag{18}$$

$$\sum_{t \in \mathcal{T} \setminus t_D} y_{se,t} \leq u_{se} \ \forall se \in \mathcal{SE} \tag{19}$$

$$\sum_{t \in \mathcal{T} \setminus t_D} y_{se,t} + \sum_{r \in er \setminus r_D} w_{se,er,r} \geq u_{se} \ \forall se \in \mathcal{SE}, er \in se, PA_{er} = 0 \tag{20}$$

Constraint (20) is necessary to ensure events are not set as active, even though they do not meet the required criteria.

The duration of active sub-events for a given event must be exactly the same as the total duration of the event (by definition of a valid XHSTT solution),

$$\sum_{se \in e} D_{se} \cdot u_{se} = D_e \ \forall e \in \mathcal{E} \tag{21}$$

A number of constraints require that the value of a deviation $V \in \mathbb{N}$ should be within an upper-limit $\overline{B}_c \in \mathbb{N}$ and a lower-limit $\underline{B}_c \in \mathbb{N}$. This means that the penalty is defined as the amount which the value of a deviation exceeds $\overline{B}_c$ or falls short of $\underline{B}_c$. To simplify notation for these cases, we introduce the function $\mathcal{U}_{\underline{B}_c, \overline{B}_c} V$, which is defined as follows:

$$s \geq \mathcal{U}_{\underline{B}_c, \overline{B}_c} V \Rightarrow \begin{cases} s \geq V - \overline{B}_c \\ s \geq \underline{B}_c - V \end{cases} \tag{22}$$

Thereby the slack-variable s is forced to take the actual value of the imposed penalty.

A resource is *busy* at some time if it attends at least one solution event at that time, and busy at some time group if it is busy at one or more times within times of that time group. Let variable $q_{r,t} \in \{0,1\}$ take value 1 if resource $r \in \mathcal{R}$ is busy in time $t \in \mathcal{T}$, and 0 otherwise. Similarly, let the binary variable $p_{r,tg} \in \{0,1\}$ take value 1 if resource $r \in \mathcal{R}$ is busy in time group $tg \in \mathcal{TG}$, and 0 otherwise. The values of the two variables are determined by the following constraints.

$$|\mathcal{SE}| \cdot q_{r,t} \geq v_{t,r} \ \forall r \in \mathcal{R}, t \in \mathcal{T} \setminus t_D \tag{23}$$

$$q_{r,t} \leq v_{t,r} \ \forall r \in \mathcal{R}, t \in \mathcal{T} \setminus t_D \tag{24}$$

$$p_{r,tg} \geq q_{r,t} \ \forall r \in \mathcal{R}, tg \in \mathcal{TG}, t \in tg \tag{25}$$

$$p_{r,tg} \leq \sum_{t \in tg} q_{r,t} \ \forall r \in \mathcal{R}, tg \in \mathcal{TG} \tag{26}$$

Constraints (23) and (25) establishes lower bounds for the variables $q_{r,t}$ and $p_{r,tg}$, i.e. ensures that these must take value 1 in case the resource is actually busy in the respective time/time group. Constraints (24) and (26) are necessary to ensure that in case the resource is in fact *not* busy in the respective time/time group, variables $q_{r,t}$ and $p_{r,tg}$ must take value 0.

In the following the constraint types of the XHSTT documentation are formulated one by one. Each constraint type is described in brief terms, and we refer to Kingston (2013b) for more details. The formulation of these constraints in terms of a Mixed-Integer Linear Programming model has not been published before. We let the 'pseudo-set' $\bar{\mathcal{C}} \subseteq \mathcal{C}$ denote constraints of a certain type depending on the context, for instance the set of all *assign resource* constraints. Furthermore we in the following make use of the general slack variable $s_{c,p,d}$, and will for each type of constraint implicitly define a corresponding slack variable with the appropriate indices for point-of-applications and deviations.

3.2.2 Assign Resources

Applies to: Events
Point-of-application: Event-resource
An *Assign Resource* constraint penalizes event resources that are not assigned resources. Specifically, the deviation at one point of application (an event resource with the appropriate role) is the sum of the duration of the sub-events of the respective event which are not assigned a resource. The cost of this constraint is given by:

$$D_e - \sum_{\substack{se \in e \\ r \in er \setminus r_D}} D_{se} \cdot w_{se,er,r} = s_{c,er}^{\text{assignres}} \quad \begin{array}{l} \forall c \in \bar{\mathcal{C}}, e \in c, \\ er \in e, \\ \text{role}_{er} = \text{role}_c \end{array} \tag{27}$$

3.2.3 Assign Time

Applies to: Events
Point-of-application: Events
The assign time constraint penalizes sub-events which are not assigned times. The deviation at one point of application is the total duration of those sub-events derived from a specific event that are not assigned a time.

$$D_e - \sum_{\substack{t \in \mathcal{T} \setminus t_D \\ se \in e}} D_{se} \cdot y_{se,t} = s_{c,e}^{\text{assigntime}} \ \forall c \in \bar{\mathcal{C}}, e \in c \tag{28}$$

3.2.4 Split Events

Applies to: Events
Point-of-application: Events
A *Split Event* constraint places limits on the number of sub-events that may be derived from a given event, and on their duration. Let the parameters $\underline{B}_c^{\text{amount}} \in \mathbb{N}$ and $\overline{B}_c^{\text{amount}} \in \mathbb{N}$ denote the minimum and maximum amount of sub-events which is used for a given event, respectively. And let $\underline{B}_c^{\text{dur}} \in \mathbb{N}$ and $\overline{B}_c^{\text{dur}} \in \mathbb{N}$ be the minimum and maximum duration a sub-event can have for a given event, respectively.

The cost of this constraint is given by the number of sub-events whose duration is less than $\underline{B}_c^{\text{dur}}$ or greater than $\overline{B}_c^{\text{dur}}$, and the amount by which the number of sub-events fall short of $\underline{B}_c^{\text{amount}}$ or exceed $\overline{B}_c^{\text{amount}}$. The following constraints are imposed:

$$\mathcal{U}_{\underline{B}_c^{\text{amount}}, \overline{B}_c^{\text{amount}}} \sum_{se \in e} u_{se} \leq s_{c,e}^{\text{spliteventamount}} \ \forall c \in \bar{\mathcal{C}}, e \in c \tag{29}$$

$$\sum_{\substack{se \in e \\ \underline{B}_c^{\text{dur}} > D_{se} \vee \overline{B}_c^{\text{dur}} < D_{se}}} u_{se} = s_{c,e}^{\text{spliteventdur}} \ \forall c \in \bar{\mathcal{C}}, e \in c \tag{30}$$

The full deviation for constraint $c \in \bar{\mathcal{C}}$ and event $e \in c$ is given by $s_{c,e}^{\text{spliteventdur}} + s_{c,e}^{\text{spliteventamount}}$.

3.2.5 Distribute Split Event

Applies to: Events
Point-of-application: Events
The *Distribute Split Event* constraints set limits on the number of sub-events which may be derived from an event. Let $D_c \in \mathbb{N}$ be the duration of the sub-events for which this constraint applies, and let $\underline{B}_c$ and $\overline{B}_c$ be the minimum and maximum number of sub-events of duration D_c which may be derived from a given event.

$$\mathcal{U}_{\underline{B}_c, \overline{B}_c} \sum_{\substack{se \in e \\ D_{se} = D_c}} u_{se} \leq s_{c,e,er}^{\text{distsplitevent}} \ \forall c \in \bar{\mathcal{C}}, e \in c \tag{31}$$

3.2.6 Prefer Resources

Applies to: Events
Point-of-application: Event-resources
This constraint defines that an event resource has different preferences for certain resources. The deviation is calculated by taking all the solution resources derived from the event resource that are assigned a resource that is not one of the preferred resources, and summing the duration of the sub-events that those resources lie in. Let $r \in c$ denote a preferred resources.

$$\sum_{\substack{se \in e \\ r \notin c, r \neq r_D}} D_{se} \cdot w_{se,er,r} = s_{c,er}^{\text{preferres}} \quad \begin{array}{l} \forall c \in \bar{\mathcal{C}}, e \in c, \\ er \in e, PA_{er} = 0, \\ \text{role}_{er} = \text{role}_c \end{array} \tag{32}$$

3.2.7 Prefer Times constraints

Applies to: Events
Point-of-application: Events
Like the Prefer Resources constraint, events might also have preferences for certain times. The deviation is calculated for each event by summing the duration of all sub-events which is assigned a time which is not one of the preferred time. The constraint has an optional duration-property, denoted $D_c \in \mathbb{N}_0$. If this property is given, only sub-events of duration D_c are considered. Let $t \in c$ denote a preferred time.

$$\sum_{\substack{se \in e \\ t \notin c, t \neq t_D \\ D_c = D_{se}}} D_{se} \cdot y_{se,t} = s_{c,e}^{\text{prefertime}} \ \forall c \in \bar{\mathcal{C}}, e \in c \tag{33}$$

3.2.8 Avoid Split Assignments

Applies to: Evengroups
Point-of-application: Eventgroups
Each solution resource can only have one resource assigned. However, when an event is split into sub-events, each of its event resources is split into several solution resources, and a different resource may be assigned to each of these solution resources. This constraint penalizes the assignment of different resource to these solution resources. The constraint examines all the solution resources derived from those event resources, and calculates the number of distinct resources assigned to them, ignoring unassigned solution resources. The deviation is the amount by which this number exceeds 1. Let variable $k_{c,eg,r} \in \{0,1\}$ take value 1 if event e is assigned to resource r with respect to avoid split assignment constraint c, and 0 otherwise.

$$\sum_{\substack{er \in e, PA_{er}=0 \\ \text{role}_c = \text{role}_{er}}} w_{se,er,r} \leq k_{c,eg,r} \quad \begin{array}{l} \forall c \in \bar{\mathcal{C}}, eg \in c, \\ e \in eg, se \in e \end{array} \tag{34}$$

$$\sum_{r \in \mathcal{R}} k_{c,eg,r} - 1 \leq s^{\text{avoidsplitass}}_{c,eg} \quad \forall c \in \bar{\mathcal{C}}, eg \in c \tag{35}$$

3.2.9 Spread Events

Applies to: Eventgroups
Point-of-application: Eventgroups
The *Spread Event* constraint has a deviation for each time group $tg \in c \in \bar{\mathcal{C}}$. Let $\underline{B}_{c,tg}$ and $\overline{B}_{c,tg}$ be the minimum and maximum number of sub-events of a given event which can be placed in time group tg of constraint c, respectively. The deviation for each time group is given by the amount of which the number of sub-events for the given event which fall short of $\underline{B}_{c,tg}$ or exceeds $\underline{B}_{c,tg}$.

$$\mathcal{U}_{\underline{B}_{c,tg},\overline{B}_{c,tg}} \sum_{\substack{se \in e \in eg \\ t \in tg}} y_{se,t} \leq s^{\text{spreadevent}}_{c,eg,tg} \quad \begin{array}{l} \forall c \in \bar{\mathcal{C}}, \\ eg \in c, \\ tg \in c \end{array} \tag{36}$$

3.2.10 Link Events

Applies to: Eventgroups
Point-of-application: Eventgroups
A *Link Event* constraint specifies that some events should be assigned the same times. For each event of a given event group we build the set of times that the sub-events derived from that event are running (not just starting times). The deviation is then the number of times that appear in at least one of these sets but not in all of them. Let variable $o_{e,t} \in \{0,1\}$ take value 1 if at least one sub-event of event $e \in c \in \bar{\mathcal{C}}$ is assigned to time $t \in \mathcal{T}$, and 0 otherwise. Let variable $l_{eg,t} \in \{0,1\}$ take value 1 if at least one event of event group $eg \in c$ is assigned to time $t \in \mathcal{T}$, and 0 otherwise. Constraints (37) and (39) ensure that these variables take correct values. The slack of Link Events constraints is defined in (40). Constraint (38) is necessary to restrict $o_{e,t}$ to take value 1 in cases where the event is in fact not assigned to the particular time, which would avoid the penalty given by constraint (40), if any.

$$\sum_{t' \in T^{\text{start}}_{se,t}} y_{se,t'} \leq o_{e,t} \quad \forall e \in \mathcal{E}, se \in e, t \in \mathcal{T} \setminus t_D \tag{37}$$

$$\sum_{\substack{se \in e \\ t' \in T^{\text{start}}_{se,t}}} y_{se,t'} \geq o_{e,t} \quad \forall e \in \mathcal{E}, t \in \mathcal{T} \setminus t_D \tag{38}$$

$$l_{eg,t} \geq o_{e,t} \quad \forall eg \in \mathcal{EG}, e \in eg, t \in \mathcal{T} \setminus t_D \tag{39}$$

$$l_{eg,t} - o_{e,t} \leq s^{\text{linkevent}}_{c,eg,t} \quad \begin{array}{l} \forall c \in \bar{\mathcal{C}}, eg \in c, e \in eg, \\ t \in \mathcal{T} \setminus t_D \end{array} \tag{40}$$

3.2.11 Order Events

Applies to: Pair of events
Point-of-application: Pair of events
An *Order Event* constraint specifies that the times two events are assigned should be in order, such that the first event ends before the second event starts. Let the parameters $\underline{B}_c \in \mathbb{N}$ and $\overline{B}_c \in \mathbb{N}$ be the minimum and maximum number of times that may separate the two events, respectively. Let $(e, e') \in c$ denote an *EventPair* which this constraint applies to. Let the variable $h^{\text{last}}_e \in \mathbb{N}$ be the ordinal number of the latest time assigned to any sub-event of event e. Let the variable $h^{\text{first}}_e \in \mathbb{N}$ be the ordinal number of the first assigned to any sub-event of event e'. The deviation is then given by the amount by which the difference between these two numbers exceeds $\overline{B}_c$ or falls short of $\underline{B}_c$.

$$\rho(t) \cdot y_{se,t} + D_{se} \leq h^{\text{last}}_e \quad \begin{array}{l} \forall c \in \bar{\mathcal{C}}, e \in c, \\ se \in e, t \in \mathcal{T} \end{array} \tag{41}$$

$$|\mathcal{T}| - (|\mathcal{T}| - \rho(t)) \cdot y_{se,t} \leq h^{\text{first}}_e \quad \begin{array}{l} \forall c \in \bar{\mathcal{C}}, e \in c, \\ se \in e, t \in \mathcal{T} \end{array} \tag{42}$$

$$\mathcal{U}_{\underline{B}_c,\overline{B}_c}(h^{\text{last}}_{e'} - h^{\text{first}}_e) \leq s^{\text{orderevents}}_{c,(e,e')} \quad \begin{array}{l} \forall c \in \bar{\mathcal{C}}, \\ (e, e') \in c \end{array} \tag{43}$$

3.2.12 Avoid Clashes

Applies to: Resources
Point-of-application: Resources
These constraints specify that certain resources should have no clashes, i.e. they should not be assigned two or more events simultaneously. The constraint produces a set of deviations at each point of application (each resource). For each time a resource is assigned two or more solution resource, there is one deviation with a value equal to the number of solution resources minus one.

$$v_{t,r} - 1 \geq s^{\text{avoidclashes}}_{c,r,t} \ \forall c \in \bar{\mathcal{C}}, r \in c, t \neq t_D \tag{44}$$

3.2.13 Avoid Unavailable Times

Applies to: Resources
Point-of-application: Resource
An *Avoid Unavailable Times* constraint specifies that certain resources are unavailable for all events at certain times. The deviation is the number of unavailable times during which the resource attends at least one solution event. $t \in c$ denotes that t is an unavailable time for constraint $c \in \bar{\mathcal{C}}$.

$$\sum_{t \in c} q_{r,t} = s^{\text{unavailabletimes}}_{c,r} \ \forall c \in \bar{\mathcal{C}}, r \in c \tag{45}$$

3.2.14 Limit Idle Times

Applies to: Resources
Point-of-application: Resources
A resource is idle at some time $t \in tg$ wrt. time group tg if it is not attending any sub-events at that time, but it is busy at some earlier time and at some later time in time group tg. The *Limit Idle Times* places limits on the number of idle times a resources may have. Let the variables $h^{\text{first}}_{r,tg} \in \mathbb{N}$ and $h^{\text{last}}_{r,tg} \in \mathbb{N}$ indicate the ordinal number of the first and the last time, respectively, where resource $r \in \mathcal{R}$ is busy in time group tg. Let $|tg|$ denote the amount of times in time group tg. Let the variable $h_{r,tg} \in \mathbb{N}$ denote the number of idle times of resource $r \in \mathcal{R}$ in time group $tg \in \mathcal{TG}$.

$$|tg| - (|tg| - \rho(t)) \cdot q_{r,t} \geq h^{\text{first}}_{r,tg} \ \forall r \in \mathcal{R}, tg \in \mathcal{TG}, t \in tg \tag{46}$$

$$\rho(t) \cdot q_{r,t} \leq h^{\text{last}}_{r,tg} \ \forall r \in \mathcal{R}, tg \in \mathcal{TG}, t \in tg \tag{47}$$

$$h^{\text{last}}_{r,tg} - h^{\text{first}}_{r,tg} + 1 - \sum_{t \in tg} q_{r,t} = h_{r,tg} \ \forall r \in \mathcal{R}, tg \in \mathcal{TG} \tag{48}$$

For each resource of the constraint the deviation is calculated as follows. Calculate the total amount of idle times for all times $tg \in c$, and find the amount which this summed value falls short of minimum $\underline{B}_c \in \mathbb{N}$ or exceeds maximum $\overline{B}_c \in \mathbb{N}$. The deviation is then given by the sum of these amounts.

$$\mathcal{U}_{\underline{B}_c, \overline{B}_c} \sum_{tg \in c} h_{r,tg} \leq s^{\text{idletimes}}_{c,r} \ \forall c \in \bar{\mathcal{C}}, r \in c \tag{49}$$

3.2.15 Cluster Busy Times

Applies to: Resources
Point-of-application: Resources
A *Cluster Busy Times* constraint limits the number of time groups during which a resource may be busy. The deviation is given by the amount of by which the number of given time groups during which the resource is busy falls short of minimum, $\underline{B}_c \in \mathbb{N}$, or exceeds maximum, $\overline{B}_c \in \mathbb{N}$. Let $tg \in c$ denote a time group which this constraint applies to.

$$\mathcal{U}_{\underline{B}_c, \overline{B}_c} \sum_{tg \in c} p_{r,tg} \leq s^{\text{clusterbusy}}_{c,r} \ \forall c \in \bar{\mathcal{C}}, r \in c \tag{50}$$

3.2.16 Limit Busy Times

Applies to: Resources
Point-of-application: Resources
The *Limit Busy Times* constraints places limits on the number of times a resource may be busy within some time groups. These constraints produces a set of deviation at each point-of-application, one for each given time group. The deviations are given by the amount by which the number of times of the given time group that the resource is busy falls short of minimum, $\underline{B}_c \in \mathbb{N}$, or exceeds maximum $\overline{B}_c \in \mathbb{N}$.

$$-|tg| \cdot (1 - p_{r,tg}) + \mathcal{U}_{\underline{B}_c, \overline{B}_c} \sum_{t \in tg} q_{r,t} \leq s^{\text{limitbusy}}_{c,r,tg} \ \forall c \in \bar{\mathcal{C}}, r \in c, tg \in c \tag{51}$$

3.2.17 Limit Workload

Applies to: Resources
Point-of-application: Resources
A workload of a solution resource is given by $W_{e,se,er} = \frac{D_{se} \cdot L_{er}}{D_e}$, where $L_{er} \in \mathbb{N}$ is the workload of event resource er. The value is a floating-point number. A *Limit Workload Constraint* places limits on the total workload of solutions resources that certain resources are assigned to. The deviation of this constraint is the amount by

which the total workload of the solution resources assigned to that resource falls short of $\underline{B}_c \in \mathbb{N}$ or exceeds $\overline{B}_c \in \mathbb{N}$, rounded up to the nearest integer.

$$\mathcal{U}_{\underline{B}_c,\overline{B}_c} \sum_{\substack{e\in c, t\in\mathcal{T}\setminus t_D \\ se\in e, er\in e}} W_{e,se,er} \cdot x_{se,t,er,r} \leq s^{\text{limitworkload}}_{c,r} \quad \forall c \in \bar{\mathcal{C}}, r \in c \qquad (52)$$

3.3 Mixed-Integer Programming Model

Given the definitions of all constraint types of XHSTT, and their respective slack variables, the objective of the model can be stated as eq. (53), setting aside the fact that some constraints are hard-constraints and some are soft-constraints.

$$\begin{aligned} z = & f(s^{\text{assignres}}_{c,er}) + f(s^{\text{assigntime}}_{c,e}) \\ & + f(s^{\text{spliteventamount}}_{c,e} + s^{\text{spliteventdur}}_{c,e}) \\ & + f(s^{\text{distsplitevent}}_{c,e,er}) + f(s^{\text{preferres}}_{c,er}) + f(s^{\text{prefertime}}_{c,e}) \\ & + f(s^{\text{avoidsplitass}}_{c,eg}) + f(s^{\text{spreadevent}}_{c,eg,tg}) + f(s^{\text{linkevent}}_{c,eg,t}) \\ & + f(s^{\text{orderevents}}_{c,(e,e')}) + f(s^{\text{avoidclashes}}_{c,r,t}) \\ & + f(s^{\text{unavailabletimes}}_{c,r}) + f(s^{\text{idletimes}}_{c,r}) + f(s^{\text{clusterbusy}}_{c,r}) \\ & + f(s^{\text{limitbusy}}_{c,r,tg}) + f(s^{\text{limitworkload}}_{c,r}) \end{aligned} \qquad (53)$$

The full MIP would therefore consists of minimizing z, subject to eqs. (12) to (52). However, we take a different approach, as described in the next section.

3.4 Solution Approach

Even though it would be natural to simply input the MIP to a generic solver, a different approach is taken, which takes advantage of the XHSTT objective function. In this approach, the model is solved in two steps, denoted Step 1 and Step 2 in the following.

By the definition of the XHSTT objective, hard constraints always take priority over soft constraints. Therefore the following approach is taken for solving the model: In Step 1, a MIP is build which only contains the hard constraints. This MIP is given as input to the MIP solver, which is ran until the given time limit is reached, or until the model is solved to optimality. The found objective value is the hard cost of the solution. In case the time limit is reached, all variables are fixed to their final value (i.e. the value they take in the best found solution), and all the soft constraints are added to identify the true cost of the found solution. In case the MIP is solved to optimality, Step 2 is performed: All soft constraints are added and the solution process is warm-started from its previous state, with the time limit set to what remains of the original time limit. Furthermore a constraint is added which ensures that the optimal value of the hard cost is kept. Let z^{hard} denote the sum of all slack variables belonging to the hard constraints. The following constraint is added:

$$z^{\text{hard}} = \text{hard cost} \qquad (54)$$

Now this extended MIP model is solved. The cost of the obtained solution, minus the hard cost found in Step 1, is the value of the soft cost. Notice that the nature of this solution method resembles *lexicographic multi-objective optimization*.

This approach takes advantage of the capability of MIPs to issue certificates of optimality. By this we mean that focus is put on the hard constraints until a solution is found with the optimal hard cost, and then we switch focus and consider the entire problem instance. If a heuristic solution method was used the inevitable question would be: When has sufficient effort been put into minimizing the hard cost?

4 Computational Results

This section presents computational results of the developed exact method, and has two primary intentions:

- How does the MIP compete with the heuristics of the ITC2011 round 2? Thereby the potential of this MIP approach can be evaluated on fair terms with well-performing heuristics. This is the subject of Section 4.1.
- Are we able to improve the best-known solutions for some instances, or even solve them to optimality? See Section 4.2.

All tests were run on a machine with an Intel i7 CPU clocked at 2.80 GHz and 12GB of RAM, running Windows 8 64 bit. In all cases the commercial state-of-art MIP solver Gurobi 5.5.0 was used. Two distinct sets of XHSTT instances have been used, both obtained from the XHSTT website (Post, 2013a). All obtained solutions have been verified as being valid using the evaluator *HSEval* (Kingston, 2013c).

As described in Section 3.1, an XHSTT objective consists of both a hard cost, and a soft cost, usually denoted (hard cost, soft cost). In case a solution has a hard cost of value 0, the objective is simply written as the soft cost, as is usually done in context of the XHSTT format.

As discussed in Section 3.1, the size of the sets $\mathcal{I}$ and $\mathcal{J}$ must be selected sufficiently high. Notice further that if the size of these sets is selected high, it

can have a big impact on the amount of variables in the model. It would be possible to select these sizes based on the properties of the constraints having Cost-Function `SumSquare` or `SquareSum`, however this is a quite complex operation as it must be derived based on each constraint-type. Instead we have selected $|\mathcal{I}| = 10$ $|\mathcal{J}| = 10$, such that the maximum possible penalty is $9^2 = 81$. This means that we cannot claim optimality for solutions with objective > 81. An easy fix for this issue is to simply perform a re-run of Gurobi if a solution is claimed as optimal, with the size of the sets set to a higher value. The same is applicable for lower bounds of value > 81. We consider this is an implementation detail, and it will be seen that in practice it has no impact of the obtained results.

4.1 International Timetabling Competition 2011

This section compares the exact method with the results obtained by the finalists in Round 2 of ITC2011. In this round the solver for each participating team was tested on 18 previously unknown instances from the archive `XHSTT-ITC2011-hidden`. The time limit for all instances was nominated to 1000 seconds, but the organizers provided a tool to benchmark machines to find the machine-dependent equivalent of this time limit. On our machine this amended to 772 seconds. The possibility to benchmark machines facilitates a fair comparison with the competitors of ITC2011, except for the fact that the rules of ITC2011 did not allow the use of commercial software, which conflicts with our use of Gurobi. The aim of this section is therefore to demonstrate the potential of MIP in the context of timetabling (which is often overlooked), and not to claim how our approach would have positioned itself in ITC2011.

In terms of solver parameters, default settings are used, except for the *pseudo-parameter* `MIPFocus` which is set to value 1, emphasizing that we are mainly interested in finding incumbent solutions. Gurobi was only allowed to use a single CPU thread, as specified in the rules of ITC2011.

The participants of ITC2011 round 2 ran their algorithm 10 times on each instance, to eliminate the stochastic impact on the results. Since we are interested in the average performance of each participant for comparison, the following processing of the results was performed: For each instance and each participant, calculate both the average hard cost and the average soft cost, and round both to nearest integer. These numbers then denote how this participant performed on this instance.

Table 2 shows the obtained results. The value of "Avg. Ranking" was calculated as follows. Each solution method was ranked 1 to 5 on each instance, 1 being the best, and the average of these ranks was taken. According to this measure, the exact method of this paper is competitive with the methods used at ITC2011. Notice in particular that the exact method performs well on the smaller instances, and is generally not as competitive on the larger instances. On three instances the exact method gave the best results.

4.2 Aiming at Optimality

In attempt to produce new (optimal) solutions, the XHSTT archive `ALL_INSTANCES` was used, which contains 38 non-artificial instances. According to the website, this archive "contains all latest versions of the contributed instances". For 10 of the instances, a solution with cost 0 is already known, which constitutes an optimal solution by the definition of XHSTT. Hence these instances are skipped in this test. Notice that `ALL_INSTANCES` contains instances which originally came from `XHSTT-ITC2011-hidden`, but due to bug-fixes in some of the instances, we consider them as two separate sets of instances (by bug-fixes we mean altering of certain constraints, such that objective values are incomparable). We refer to (Post, 2013a) for instance-statistics.

This test was performed with the following setup: Gurobi is allowed to use all CPU cores (which is 8 in our case), and the time-limit is set to 24 hours for each instance. As initial solution for each instance, the current best known solution is provided. Default parameter settings of Gurobi were used. Table 3 shows the obtained results. A gap between an incumbent solution x and a lower bound LB is calculated by $\frac{|x-\mathrm{LB}|}{x}$.

For each instance a solution with XHSTT objective (H, S) is found, as well as a lower bound $(\underline{H}, \underline{S})$. By the definition of our solution method, we only have a lower bound on the soft cost $\underline{S}$ iff an optimal solution for the hard cost is known, i.e. $H = \underline{H}$. If a lower bound or an objective value is not found we write "-". Notice that even though we give the current best known solution as starting solution, Gurobi might still not find a solution for Step 1, usually in case the instance in question is of huge size. In Table 3, both the gap for the hard cost and the soft cost is shown (in case the required costs and lower bounds are available).

Table 3 shows that our method obtains better solutions for 8 instances. 4 instances was solved to optimality, proving optimality of 3 previously known solutions and finding 1 new optimal solution. Furthermore, 11 new non-trivial lower bounds and 7 new best solutions have been established for the instances which were not solved to optimality.

Table 2 Performance of the MIP using same running time as specified in ITC2011. For each instance is listed the average solution found from each of the competitors of ITC2011, and the solution obtained by the MIP formulations. The best solutions are marked in **bold**. Objectives marked with * are optimal solutions.

Instance	GOAL	HySST	Lectio	HFT	Exact method
BrazilInstance2	(1, 62)	(1, 77)	**38**	(6, 190)	46
BrazilInstance3	124	118	152	(30, 283)	**39**
BrazilInstance4	(17, 98)	(4, 231)	**(2, 199)**	(67, 237)	(5, 286)
BrazilInstance6	(4, 227)	(3, 269)	**230**	(23, 390)	682
ElementarySchool	4	(1, 4)	**3**	(30, 73)	**3**
SecondarySchool2	**1**	23	34	(31, 1628)	(1604, 3878)
Aigio	**13**	(2, 470)	1062	(50, 3165)	(1074, 3573)
Italy_Instance4	**454**	6926	651	(263, 6379)	17842
KosovaInstance1	**(59, 9864)**	(1103, 14890)	(275, 7141)	(989, 39670)	(3626, 2620)
Kottenpark2003	**90928**	(1, 56462)	(50, 69773)	(209, 84115)	(8491, 6920)
Kottenpark2005A	**(31, 32108)**	(32, 30445)	(350, 91566)	(403, 46373)	(2567, 53)
Kottenpark2008	**(13, 33111)**	(141, 89350)	(209, 98663)	-	(14727, 5492)
Kottenpark2009	**(28, 12032)**	(38, 93269)	(128, 93634)	(345, 99999)	(17512, 140)
Woodlands2009	(2, 14)	(2, 70)	**(1, 107)**	(62, 338)	(1801, 705)
Spanish school	**894**	1668	2720	(65, 13653)	(1454, 11020)
WesternGreece3	**6**	11	(30, 2)	(15, 190)	25
WesternGreece4	**7**	21	(36, 95)	(237, 281)	81
WesternGreece5	**0**	4	(4, 19)	(11, 158)	15
Avg. Ranking	1.72	2.67	2.50	4.44	3.61

4.2.1 Alternative Formulation

The Limit Idle Times constraint is known to be difficult for solvers to handle (Dorneles et al (2012)). In our formulation, this constraint is formulated using Big-M notation (constraints (46) and (47)), which can provide bad LP-relaxation, which in turn might slow down the solution process. Furthermore this constraint is part of most instances (29 of 38 instances in the `ALL_INSTANCES` archive), so an alternate formulation is proposed. The alternate formulation uses variable $h_{r,tg,t} \in \{0,1\}$ which takes value 1 if resource $r \in \mathcal{R}$ has an idle time in time $t \in tg$ in time group tg, and 0 otherwise. Constraints (46), (47) and (48) are replaced by

$$q_{r,t'} - q_{r,t} + q_{r,t''} - 1 \leq h_{r,tg,t} \quad \begin{array}{l} \forall r \in \mathcal{R}, tg \in \mathcal{TG}, \\ t, t', t'' \in tg, \\ \rho(t') < \rho(t) < \rho(t'') \end{array} \tag{55}$$

This yields more rows in the MIP; for each time group $tg \in \mathcal{TG}$ the amount of additional constraints is $\binom{|tg|}{3}$. Furthermore, there is a great possibility that the amount of variables increases due to the extra dimension on the h variable. However, no Big-M notation is used.

Due to the possible big increase in the size of the model, this alternative formulation is only tested on the smaller instances from archive `ALL_INSTANCES`, skipping those instances in which the optimal solution was found in the previous test (Table 3). Since the goal is to achieve is good solutions as possible, we restart the procedure from the best found solution of Table 3 and run it for additional 24 hours. This test-setup means that we cannot compare the performance of the two formulations. Table 4 shows the obtained results. The table shows that this formulation is capable of finding 2 new optimal solutions. For the instances not solved to optimality, 6 lower bounds were improved, and new best solutions were found for 6 instances.

5 Conclusion

This paper has shown the first exact method for (High) School Timetabling instances in the XHSTT format. A solution method which takes advantage of the structure of the objective function of XHSTT has been proposed. For the most recent version of the archive `ALL_INSTANCES`, we were able to produce 2 new optimal solutions and prove optimality of 4 previously known solutions. For 11 other instances, new non-trivial lower bounds were shown. For the instances not solved to optimality, we were able to improve the best known solution in 9 cases.

Table 3 Performance of the MIP on `ALL_INSTANCES`. For each instance is listed the best previously known solution *"Best"*, and for the solution found by our approach is listed the time used to solve Step 1 *"$Time_1$"*, the time used to solve Step 2 *"$Time_2$"*. *"Time"* indicates the total solving time. All times have seconds as unit. Furthermore the objective *"Obj"* and the lower bound *"LB"* is listed. The percentage gap between the objective and the lower bound is divided into the gap for the hard constraints *"Gap_1"* and the gap for the soft constraints, *"Gap_2"*. Objectives in **bold** denote new best solution while optimal solutions are marked with *.

			MIP solution method						
	Instance	Best	$Time_1$	$Time_2$	Time	Obj	LB	Gap_1	Gap_2
AU	BGHS98	(3, 494)	>86400	-	>86400	(3, 494)	(-,-)	-	-
AU	SAHS96	(8, 52)	>86400	-	>86400	(8, 52)	(-,-)	-	-
AU	TES99	(1, 140)	>86400	-	>86400	(1, 140)	(0,-)	100.0	-
BR	Instance1	42	0	>86400	>86400	**40**	28	0.0	30.0
BR	Instance2	5	1	>86399	>86400	5	1	0.0	80.0
BR	Instance3	47	1	>86399	>86400	**26**	19	0.0	26.9
BR	Instance4	78	1	>86399	>86400	**61**	42	0.0	31.2
BR	Instance5	43	1	>86399	>86400	**30**	10	0.0	66.7
BR	Instance6	60	1	>86399	>86400	60	14	0.0	76.7
BR	Instance7	122	1	>86399	>86400	122	22	0.0	82.0
DK	Falkoner2012[1]	(2, 23705)	>86400	-	>86400	(2, 23705)	(0,-)	100.0	-
DK	Hasseris2012[2]	(293, 32111)	>86400	-	>86400	(293, 32111)	(-,-)	-	-
DK	Vejen2009[3]	(20, 18966)	>86400	-	>86400	(20, 18966)	(2,-)	90.0	-
UK	StPoul	136	52	>86348	>86400	136	0	0.0	100.0
FI	ElementarySchool	3	2	785	787	*3	3	0.0	0.0
FI	HighSchool	1	1	>86399	>86400	1	0	0.0	100.0
FI	SecondarySchool	88	1	>86399	>86400	88	77	0.0	12.5
GR	UniInstance3[4]	5	0	3	3	*5	5	0.0	0.0
GR	UniInstance4[5]	8	1	>86399	>86400	8	0	0.0	100.0
IT	Instance1	12	1	4561	4562	*12	12	0.0	0.0
IT	Instance4	78	12	>86389	>86400	**62**	27	0.0	56.5
XK[6]	Instance1	3	31	>86369	>86400	3	0	0.0	100.0
NL	GEPRO	(1, 566)	>86400	-	>86400	(1, 566)	(0,-)	100.0	-
NL	Kottenpark2003	1410	57	>86343	>86400	1410	(0,-)	0.0	-
NL	Kottenpark2005	1078	88	>86312	>86400	1078	9	0.0	99.2
NL	Kottenpark2009	9250	92	>86308	>86400	**9035**	160	0.0	98.2
ZA	Woodlands2009	2	22	77878	77900	***0**	0	0.0	0.0
ES	School	(3, 5966)	6525	>79875	>86400	**357**	322	0.0	9.8

[1] Shorthand for instance *FalkonergaardenGymnasium2012*
[2] Shorthand for instance *HasserisGymnasium2012*
[3] Shorthand for instance *VejenGymnasium2009*
[4] Shorthand for instance *WesternGreeceUniversityInstance3*
[5] Shorthand for instance *WesternGreeceUniversityInstance4*
[6] Kosova.

Establishing optimal solutions and lower bounds is indeed a step forward for research within high school timetabling, and for the XHSTT format in particular. This gives researchers a possibility to compare their obtained solutions with an (optimal) lower bound, which is valuable for evaluating the quality of solutions.

As subjects for future research the following are mentioned. The MIP could be used in context of *Two-Stage Decomposition* (TSD), by first assigning times to events, and secondly assigning resources to event resources. Thereby the resource-assignments are done subject to the times assigned to events. Such an approach was used with great success in the paper of Lach and Lübbecke (2012) for the *Curriculum-based University Timetabling Problem* (the optimization problem used in the International Timetabling Competition 2007), and by Sørensen and Dahms (2013) for the real-world case of High School Timetabling in Denmark. In both of these papers, the TSD is theoretically capable of producing near-optimal results, even

Table 4 Performance of the alternative formulation on the smaller instances of archive `ALL_INSTANCES`. All the columns are defined in analogous way to Table 3, except for *"Obj_{T3}"* and *"LB_{T3}"* which denote the objective value and the lower bound found in Table 3.

					MIP alternative formulation						
	Instance	Best	Obj_{T3}	LB_{T3}	$Time_1$	$Time_2$	Time	Obj	LB	Gap_1	Gap_2
BR	Instance1	42	40	28	0	1918	1918	***38**	38	0.0	0.0
BR	Instance2	5	5	1	0	290	290	*5	5	0.0	0.0
BR	Instance3	47	26	19	1	>86399	>86400	**23**	21	0.0	8.7
BR	Instance4	78	61	42	1	>86399	>86400	61	49	0.0	19.7
BR	Instance5	43	30	10	1	>86399	>86400	**26**	15	0.0	42.3
BR	Instance6	60	60	14	1	>86399	>86400	**59**	18	0.0	69.5
BR	Instance7	122	122	22	1	>86399	>86400	**84**	26	0.0	69.1
FI	HighSchool	1	1	0	1	>86399	>86400	1	0	0.0	100.0
FI	SecondarySchool	88	88	77	1	>86399	>86400	**84**	77	0.0	8.3
GR	UniInstance4[1]	8	8	0	1	>86399	>86400	8	0	0.0	100.0
IT	Instance4	78	62	27	6	>86394	>86400	**57**	27	0.0	52.6
ES	School	(3, 5966)	357	322	44	>86356	>86400	357	330	0.0	7.6

[1] Shorthand for instance *WesternGreeceUniversityInstance4*

though the problem is split into two separate MIPs. However, the XHSTT case is possibly less suited for this type of decomposition as instances might contain a majority of constraints related to resource assignments. Since the assignments to times for events are performed in the first stage of the decomposition, and because these assignments cannot be altered when the resource-assignments are performed, a TSD approach would possibly be heuristic in nature. Obviously, if an XHSTT instance have all resources preassigned to event resources, a TSD would be unnecessary.

Our MIP formulation is exponential in size by the amount of sub-events in the instance, as all possible combinations of sub-events are generated. A better formulation would be less dependent on this amount. One could for instance solve the model iteratively, and 'inject' new sub-events in the model on-the-fly. Another possibility would be to consider a formulation which simulate sub-events by an integer variable which define the lengths of each respective active sub-event. Such improved formulations are subject for future research.

References

Avella P, D'Auria B, Salerno S, Vasilâev I (2007) A computational study of local search algorithms for italian high-school timetabling. Journal of Heuristics 13:543–556

Birbas T, Daskalaki S, Housos E (2009) School timetabling for quality student and teacher schedules. J of Scheduling 12:177–197

Bixby RE (2012) Optimization Stories, 21st International Symposium on Mathematical Programming Berlin, vol Extra, Journal der Deutschen Mathematiker-Vereinigung, chap A Brief History of Linear and Mixed-Integer Programming Computation, pp 107–121

ter Braak M (2012) A hyperheuristic for generating timetables in the xhstt format. Master's thesis, University of Twente

Dorneles ÁP, de Araújo OC, Maria-Brazil S, Buriol LS (2012) The impact of compactness requirements on the resolution of high school timetabling problem. In: Congreso Latino-Iberoamericano de Investigación Operativa

Fonseca G, Santos H, Toffolo T, Brito S, Souza M (2012) A sa-ils approach for the high school timetabling problem. In: Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012)

Kheiri A, Ozcan E, Parkes AJ (2012) Hysst: Hyperheuristic search strategies and timetabling. In: Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012), pp 497–499

Kingston J (2013a) Educational timetabling. In: Uyar AS, Ozcan E, Urquhart N (eds) Automated Scheduling and Planning, Studies in Computational Intelligence, vol 505, Springer Berlin Heidelberg, pp 91–108

Kingston JH (2013b) High school timetable file format specification: Constraints. `http://sydney.edu.au/engineering/it/~jeff/hseval.cgi?op=spec&part=constraints` [Re-

trieved 12/8-2013]
Kingston JH (2013c) The hseval high school timetable evaluator. http://sydney.edu.au/engineering/it/~jeff/hseval.cgi [Retrieved 12/8-2013]
Lach G, Lübbecke M (2012) Curriculum based course timetabling: new solutions to udine benchmark instances. Annals of Operations Research 194:255–272
Pimmer M, Raidl G (2013) A timeslot-filling heuristic approach to construct high-school timetables. In: Di Gaspero L, Schaerf A, Stützle T (eds) Advances in Metaheuristics, Operations Research/Computer Science Interfaces Series, vol 53, Springer New York, pp 143–157
Post G (2013a) Benchmarking project for (high) school timetabling. http://www.utwente.nl/ctit/hstt/ [Retrieved 12/8-2013]
Post G (2013b) International timetabling competition 2011 results. http://www.utwente.nl/ctit/hstt/itc2011/results/ [Retrieved 12/8-2013]
Post G, Ahmadi S, Daskalaki S, Kingston J, Kyngas J, Nurmi C, Ranson D (2012a) An xml format for benchmarks in high school timetabling. Annals of Operations Research 194:385–397
Post G, Gaspero LD, Kingston JH, McCollum B, Schaerf A (2012b) The third international timetabling competition. In: Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012), Son, Norway
Romrös J, Homberger J (2012) An evolutionary algorithm for high school timetabling. In: Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012), SINTEF, pp 485–488
Santos H, Uchoa E, Ochi L, Maculan N (2012) Strong bounds with cut and column generation for class-teacher timetabling. Annals of Operations Research 194(1):399–412
Schaerf A (1999) A survey of automated timetabling. Artificial Intelligence Review 13:87–127
Sørensen M, Dahms FHW (2013) A two-stage decomposition of high school timetabling applied to cases in denmark. Computers & Operations Research To appear
Sørensen M, Stidsen TR (2013) Integer programming and adaptive large neighborhood search for real-world instances of high school timetabling. Annals of Operations Research PATAT 2012 SI:Submitted Jan 21. 2013
Sørensen M, Kristiansen S, Stidsen TR (2012) International timetabling competition 2011: An adaptive large neighborhood search algorithm. In: Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012), SINTEF, pp 489–492
Valouxis C, Gogos C, Alefragis P, Housos E (2012) Decomposing the high school timetable problem. In: Practice and Theory of Automated Timetabling (PATAT 2012), Son, Norway